371

PART 2: METHODOLOGICAL CONSIDERATIONS FOR AGENT-BASED MODELING OF LAND-USE AND LAND-COVER CHANGE

Thomas Berger, Michael Goodchild, Marco A. Janssen, Steven M. Manson, Robert Najlis, and Dawn C. Parker

Part 2 synthesizes meeting discussions on a set of pre-defined topics identified by workshop organizers: the appropriate role for and potential applications of ABM/LUCC; spatial issues in model development; the availability and adequacy of software modeling tools; and issues in calibration, verification, and validation of LUCC models. For each of these topics, participants identified particular areas of concern for ABM/LUCC. As seen in sections 3.2–3.9, researchers involved in particular projects were asked to report on their current strategies for addressing each of these areas of concern. The following sections provide overviews of these areas, highlight challenges for ABM/LUCC, and consider potential solutions.

2.1. Potential Strengths and Appropriate Roles for ABM/LUCC

Roles, Scope, and Methodology of Models

A substantial portion of meeting discussions was devoted to discussion of the potential roles, scope, and methodology behind ABM/LUCC. Workshop participants proposed several interrelated continua along which ABM/LUCC could be categorized (see Figure 1).

Along with the matrix that will be developed in section 3.1, this framework provides a useful context for relating ABM/LUCC to previous LUCC modeling work. The continuum can be defined in most general terms as running from purely theoretical to purely empirical. Theoretical models are often constructed to serve as explanatory tools, and thus results are often generalizable to a range of research applications. In contrast, empirical models are often designed to closely match the details of a particular case study, and as such, their conclusions are often specific to that case. However, both theoretical and empirical models potentially can serve as exploratory tools, as discussed further below.

Theoretical models often can be characterized as deductive, in the sense that they use a logical procedure to derive some very specific results from basic and unquestioned assumptions (axioms). Inductive methods, in contrast, filter patterns from empirical data to identify some general laws behind them. Thus, in principle, the spectrum also could be seen as running from deductive to inductive. However, there is substantial debate as to whether computer simulation methods such as ABM can be characterized as purely deductive and/or purely inductive. This topic is discussed in greater detail in Parker *et al.* (2002).

In addition to this continuum, two other continua may potentially characterize LUCC models. The first is normative (describing how reality could or should be under ideal circumstances) to positive (describing links between mechanisms and outcomes without judgment as to fitness or appropriateness). The focus of this report, consistent with the bulk of meeting discussions and the approach taken by Verburg *et al.* (in press), remains on positive models.

Figure 1. Continua for Categorizing Agent-Based Models

The second continuum is simple to complex. It is important to acknowledge that this continuum is distinct from the theoretical/empirical continuum, as a theoretical model may be relatively complex and an empirical model may be quite simple. Regardless of the style of model implemented, there is an ever-present danger of building too much complexity into any model, resulting in difficulty in understanding how processes drive outcomes. This point was stressed repeatedly by many workshop participants. Casti (1997) provides an excellent discussion of an appropriate level of detail in models of complex systems.

Traditionally, agent-based models have operated at the theoretical end of this spectrum, whereas other tools used by LUCC researchers have operated at the empirical end of the spectrum. Thus, one possible role for ABM/LUCC is to provide explanatory, generalizable insights that may guide applied research efforts. However, participants noted that ABM/LUCC also may serve to bridge the gap between abstract analytical models and applied statistical models. Specifically, if causal mechanisms are explicitly represented and parameterized as closely as possible with real-world data, such models may serve in a deductive style for applied policy analysis by linking processes to possible outcomes. Such models also may provide a means to test previously abstract models against real-world data, if theoretical models form the basis for defining processes in a simulation model that is then used to generate simulated data. If patterns from the simulated data statistically match patterns in real-world data, analyzed using similar inductive techniques, then support is lent to the theoretical processes used in the simulation model. This approach is suggested by Parker, et al. (in press). Judd (1997) discusses the possibility of using regression analysis to understand the results of computational simulations in a theoretical context.

The question of how ABM/LUCC relate to statistical models of land-use and land-cover change is often posed in discussions among researchers. The question of which modeling technique may have greater explanatory power often implicitly underlies the discussions. There is not yet one definite answer, although there are some concrete ways in which these models can be and have been related. In general, participants stressed complementary, rather than competing, roles for ABM/LUCC and statistical models. A more detailed discussion of these issues is provided in the expanded version of this report by Parker *et al.* (2002).

Specific Roles for ABM/LUCC

Workshop participants identified a variety of conceptual roles for which ABM/LUCC may hold advantages over other modeling techniques. While some of these roles fit into a specific category in the characterization of the four model classes discussed by Couclelis in section 1.3, others apply broadly to all agent-based LUCC models.

Computational Laboratory

Several participants have used ABM/LUCC in a deductive style to methodically explore human-environmental interactions. As such, ABM/LUCC serve as computational laboratories that allow for thought experiments and may structure the exploration of dynamic interactions. These stylized models are potentially useful for exploring links between microlevel interactions and macro-outcomes, creating long-range theoretical models of the underlying driving forces of global phenomena, exploring systems dynamics and the implications of interactions, and examining the implications of heterogeneity among decision

makers and their environment. A powerful role for such models might be to demonstrate a counterintuitive result that runs contrary to established theory and intuition.

Integrated Modeling of Human-Environment Systems

Many participants stressed the potential of ABM/LUCC to represent the co-evolution of human/environmental systems. Because models of human decision making with models of biophysical processes can be linked through a common spatial identifier, ABM/LUCC are seen to hold substantial promise for interdisciplinary modeling. This advantage comes in large part through flexibility in scale of representation on both the agent decision and biophysical modeling side. Unlike analytical models that often rely on aggregation assumptions for mathematical tractability, ABM/LUCC can be constructed to operate at the spatial scale relevant for biophysical process models. This fine-scale representation may offer statistical advantages, since spatial aggregation of data generally implies a loss of statistical information. In general, the workshop participants expect increased exploratory power for environmental models when the influence of human decision makers is included.

Representing Complexity, Emergence, and Cross-Scale Dynamics

Representing complexity is seen as a major strength of ABM/LUCC. Extensive discussions among participants occurred around the concept of "emergence," including its definition and the possible role that emergent phenomena may play in LUCC research. The notion of emergence is a central tenet of agent-based modeling, and the search for emergence is mentioned explicitly by several of the modeling efforts noted in this report. The term emergent refers to a system having qualities that are not analytically tractable from the attributes of internal components (Baas and Emmeche 1997). Emergent phenomena exhibit structures that are not explained by lower-level dynamics and typically persist beyond the average lifetimes of entities upon which they are built (Crutchfield 1994). More intuitively, an emergent property may be defined as a macroscopic outcome resulting from synergies and interdependencies between lower-level system components.

The concept of emergence and the concept of cross-scale hierarchies are potentially related. Identifying emergence, therefore, may require understanding important cross-scale interactions and deliberately building in interactions across levels, rather than limiting modeling and analysis to a single scale. Related to this theme, the group discussed the concept that emergent properties from one level of interaction may define the units of interaction at the next highest level. While the group concluded that ABM/LUCC have potential to explicitly represent cross-scale interactions and feedbacks, both bottom-up and top-down, they agreed that this potential has been minimally exploited to date.

Among participants, some debate centered on the question of whether emergence was a property of a real-world system or simply a property of a modeled system. Further debate centered on whether or not an emergent property must be "surprising" by definition. The concept of surprise is potentially consistent with the concept of an emergent property as one that could not be predicted by examining the components of the system in isolation. However, surprise is a fundamentally subjective concept. If a phenomenon must be surprising, how can it be replicable? Is it then not emergent upon reobservation? Auyang (1998) specifically rejects the concept of surprise as a defining characteristic of emergence. The concept of surprise, though, may provide a counterfactual way of defining emergence:

a pattern whose appearance is an obvious consequence of the properties of the underlying components may not be regarded as emergent. More complete discussions of emergence and cross-scale hierarchies are provided by Parker and colleagues (Parker, *et al.* 2001; Parker, *et al.* 2002).

It was suggested that emergent properties might be recognized through the language used to describe them; if new language and/or definitions are needed to describe macro-outcomes, they are potentially emergent. Auyang (1998) discusses emergent phenomena in related terms. The group agreed that for LUCC modeling, it would be useful to focus on emergent properties that are explicitly spatial and result from human-environment interactions. Examples discussed included suburban sprawl, ecosystem functions, social norms, and paths of technology diffusion. Such emergent properties may provide targets for model validation and assessment.

Conducting Interactive Experiments

Another role identified by participants for ABM/LUCC is as a tool for conducting controlled experiments with human decision makers. Multiple goals were identified, including assessing the impacts of hypothetical institutional structures on humans' decisions and subsequent land-use change, informing construction of the agent-decision-making specifications of LUCC models, providing an interactive decision-support tool for policy makers, and promoting discussion between stakeholders that may lead to awareness of the views of other co-users of the land and facilitate group decision making.

Scenario Analysis

Participants suggested that scenario development and analysis using ABM/LUCC could supplement findings from existing LUCC research. An ABM that contains a detailed structural representation of the system under study could be used to analyze alternative scenarios that frame the range of plausible driving forces of land-use change. As outlined in Lambin *et al.* (1999: 81), different types of scenarios could be developed: normative, reference, predictive, and responsive.

The potential for ABM/LUCC as tools for extrapolation and projection was seen as a major conceptual advantage by many participants, although participants emphasized a scenario analysis or prospective role, rather than a prediction role. Because extrapolation and projection are important priorities for the LUCC community, exploration of potential complementarities between statistical models and ABM/LUCC with respect to development of projective models is an important area for further research. Lessons may be drawn from previous research using parameterized, spatially explicit models such as CA, Markov models, and mathematical programming models.

Specific Research Questions

Participants compiled a tentative list of specific proposed research topics that could be addressed with ABM/LUCC:

- Temporal and spatial diffusion of technological innovations
- Modeling the impacts of transportation and communication networks

- Scenario analysis for land-use policy and planning
- Understanding structural adjustment in agriculture in response to shifts in policy incentives
- Modeling firm location decisions, impacts on demand for public services, and subsequent feedbacks among levels of spatial organization
- Examining the sustainability of human-environment systems
- Assessing the impacts of global change on land and water resources as well as possible human adaptations

2.2. Issues in Spatially Explicit Modeling

In contrast to many historical applications of agent-based modeling, ABM/LUCC are by nature spatial models. As such, they draw on a rich methodological history that has evolved in support of other spatial research methodologies. This section summarizes the workshop presentation by Michael Goodchild and subsequent discussions among meeting participants regarding the role of space in ABM/LUCC models. (A more complete discussion by Goodchild can be found in section 2.2 of Parker *et al.* 2002.) The section offers a definition of what it means to be spatially explicit, discusses the rationale for spatially explicit modeling of land-use change processes, briefly reviews some available tools that support spatial modeling, and discusses some key issues in constructing and validating spatial models.

What does it mean to be spatially explicit?

Many disciplines use the term spatially explicit, but in different ways. An ecologist or economist might call a model spatially explicit if it recognizes two markets or habitats separated by a partial communication barrier, whereas a geographer is more likely to reject such gross lumping, and to insist that a spatially explicit model be constructed in a continuous spatial frame. Nevertheless, there seem to be some simple tests that one can apply to determine if a model is spatially explicit, or if an area of investigation demands spatially explicit modeling. Four such tests were discussed at the workshop:

1. The invariance test: A model is spatially explicit if its results are not invariant under relocation of the objects of study. In other words, a model is spatially explicit if its workings are affected by randomly moving the objects that participate in the model.
2. The representation test: A model is spatially explicit if location is included in the representation of the system being modeled, in the form of coordinates or derivative spatial properties such as distances.
3. The formulation test: A model is spatially explicit if spatial concepts such as location or distance appear directly in the model, in algebraic expressions or behavioral rules.
4. The outcome test: A model is spatially explicit if the spatial forms of inputs and outputs are different. In other words, a spatially explicit model modifies the landscape on which it operates.

Any one of these tests might be sufficient to determine whether a model is spatially explicit, and a given model might satisfy any combination of the tests.

Why be spatially explicit in modeling LUCC?

Agent-based models of LUCC are complex, representing many distinct processes. For many of these processes, space may not be relevant. For example, models of the individual choices made by actors on the landscape may be essentially invariant under relocation, such that the rules governing the behavior of a decision maker are identical wherever that decision maker is located. In other cases, location and distance are surrogates for a lack of communication or transport cost, rather than actual causal factors.

However, the processes of LUCC modify landscapes, producing fragmentation, regionalization, and other types of patterns—and these patterns are of very significant interest to policy makers. Many of our research questions in LUCC modeling are related to the spatial structures of outcomes, in part because of the importance of spatial structure in other processes for which land use is an input, such as biological conservation. Therefore, the most important reason for LUCC modeling to be spatially explicit may be related to model outputs, and thus to the outcome test outlined above. Workshop participants concurred that a LUCC model is likely to be assessed, at least in part, through the spatial patterns that it produces, and their agreement with observed spatial patterns. Several of the authors in section 3 discuss spatial patterns as macroscopic outcomes of interest. It follows that ABM/LUCC should be spatially explicit as defined above.

The Toolkit for Spatially Explicit Modeling

Workshop participants discussed the range of software tools available for spatially explicit modeling. Several tools were discussed by Goodchild and other meeting participants.

Geographic information systems (GIS) were discussed as one of the most important tools for development of ABM/LUCC. GIS can be defined as systems for input, management, analysis, and output of spatially referenced information (Longley *et al.* 2001). GIS software provides the foundation for representation and handling of spatially explicit information and makes it very easy to add a wide range of analytic, modeling, and related functions. The need for dynamic GIS functionality was recognized by meeting participants. PCRaster, an instance of a GIS designed specifically for dynamic modeling, was one tool proposed as potentially useful. (The package was developed at the University of Utrecht and is available at http://www.pcraster.nl/.) However, commercially available GIS software tends to have been designed for comparatively static applications and is not easily used as the basis for dynamic models. More work is clearly needed at a technical level in integrating agent-based modeling capabilities with GIS, an area recently reviewed by Gimblett (2002). Additional meeting discussions regarding the need to integrate ABM and GIS are summarized in section 2.3.

Cellular automata software provides another environment for spatially explicit modeling. Cellular automata are typically restricted to models that can be expressed as simple rules applied to cells in a *raster*, modifying the state of one cell based on its prior state and the prior states of its immediate neighbors.

A series of spatial modeling tools support both model development and verification and validation. Much effort in recent years has gone into the development of appropriate metrics of landscape fragmentation and related properties. Fragmentation statistics can now be readily evaluated in a GIS environment and used to test LUCC models against patterns of fragmentation on real landscapes. SpaceStat (http://www.spacestat.com) supports the analysis of spatially explicit models defined over more general, irregular geometries, with spatial lags that are the two-dimensional equivalent of temporal lags. Participants also discussed the role of geostatistics, the branch of statistics based on the theory of spatially regionalized or autocorrelated variables (see, for example, Isaaks and Srivastava 1989, Burrough and McDonnell 1998), and its specific tools, such as kriging and co-kriging. These and many other forms of analysis might be used to compare model outputs to geographic reality, and thus to approach the validation of LUCC models (as discussed in section 2.4).

Challenges in Spatially Explicit Modeling

Participants in the workshop identified several challenges that are of critical importance in spatially explicit modeling. Some of these will be familiar to anyone who has worked with spatial data, while others relate to the specific context of LUCC modeling.

Scale is important in two distinct ways in LUCC modeling: in the form of extent, or the area covered by the model, and in the form of resolution, or the level of detail inherent in the model. Extent is important because of the general property of spatial heterogeneity, or the tendency for geographic context to vary slowly but consistently across the surface of the planet. It follows that the results of any analysis or modeling will depend explicitly on the choice of study area, and that it is virtually impossible to select any area to be typical or representative of the Earth's surface as a whole, or any substantial part of it. Generalization from a single case study over a limited area is necessarily difficult. Resolution is critical because of the role it plays in determining whether a model of LUCC can be successful. Any spatially explicit process has an inherent scale, and attempts to model the process at levels of resolution coarser than the inherent scale will inevitably fail, because details that are important to the process will be missed by the model. Additional discussions of scale-related issues are provided in section 2.4 and by Quattrochi and Goodchild (1997), Gibson *et al.*(1998), Atkinson and Tate (2000), and Agarwal *et al.* (in press).

Researchers building spatial models face a series of choices as to how spatial variation is digitally represented. Choices exist in levels of detail, the objects represented in the database, whether or not to represent the third spatial dimension, whether to recognize change through time, whether to implement models using raster or *vector* representations of spatial variation, and in many other aspects. These choices are recognized in the form of alternative data models and structures. In its most general form, data modeling is the study of ontology, a branch of science concerned with the process of description. One of the most fundamental distinctions in spatially explicit ontology is between fields—descriptions that conceptualize the Earth's surface in terms of the continuous variations of measurable quantities such as elevation or temperature—and discrete objects that litter an otherwise empty space and can be counted and manipulated. Examples of the latter include biological organisms, landscape features such as lakes or habitat patches, and human constructions such as buildings.

Ontology is critical because it ultimately affects the types of models that can be built. Human agents might be conceptualized as discrete objects, moving around in space, but influenced in their behaviors by continuously varying fields capturing such properties as population density (and hence crowding), agricultural suitability, land rent, or climate. In turn, these discrete objects and fields must be represented in digital form, at levels of detail that are appropriate to the processes being modeled. The combination of mobile objects and spatial fields which are dynamically updated in response to human actions was recognized by participants as defining one of the major challenges for development of ABM/LUCC.

2.3. Software Tools and Communication Issues

This section briefly summarizes the workshop's discussions related to the design and implementation of ABM/LUCC in computer code. We can only outline here the advantages of Object-Oriented Programming (OOP), a programming technique that is typically employed for ABM/LUCC. Najlis *et al.* (section 2.3 in Parker *et al.* 2002) discuss these programming issues in much more detail using the instructive example of hypothetical land managers who interact through land markets. The section continues with summaries of meeting discussions on various software-related topics and concludes with a review of existing simulation packages designed for agent-based modeling.

OOP As an Organizing Technique

Object-oriented programming languages were developed as a means to organize code into separate concerns and manageable units. The programming works to organize code by encapsulating both data and the functions that act upon that data into one unit (a class). By combining needed data and functions into one class, OOP means to provide fairly self-sufficient units, facilitating efficient verification and updating of class functions. For example, in writing code about how a farm manager agent makes decisions, the programmer would not want to be concerned with how trees grow on the landscape. Figure 2 illustrates how a farm agent class might represent real farm-households in ABM/LUCC. This class incorporates data such as behavioral characteristics and factor endowments with functions that handle different decision-making problems. Figure 2 illustrates how this farm agent class might be embedded within an object-oriented ABM/LUCC model.

Classes are central to OOP. They can be organized and used in a number of different ways. Inheritance is useful when classes are organized into a hierarchy. In such a case, subclasses inherit data and functions from the superclass above them. For example, the farm agent might inherit from a superclass such as an agent class. Classes also can be arranged through the concept of composition. That is, one class contains or has an instance or instances of another class. For instance, the farm agent class might have an agent decision module. Composition is useful when one class frequently makes use of some code but wants to keep that code separate for some reason. The fact that the superclass and all subclasses define data and functions that serve the same roles allows the classes to be used interchangeably through polymorphism. That is, no matter whether we use the superclass or one of the subclasses, the same data and functions will be available for use. The functions and data will have the same names, will be called in the same way, and will return the same type of information (though

of course they may return different values). Polymorphism is often, though not exclusively, accomplished through inheritance. It gives a great deal of flexibility to code.

The Land Market – An Example

The use of OOP is briefly demonstrated through the following simple example of a land-market model. The example illustrates the process of putting a piece of land up for lease. The example in Figure 3 demonstrates the use of inheritance and composition. Additionally, there is a class in the center of the diagram called the Mediator class. The Mediator is an organization class that facilitates communication between other classes. This organizational structure is an example of the Mediator pattern. There are many useful patterns that can be used to design OOP. A good, though perhaps advanced, resource on this topic is Gamma *et al.* (1995). The Landscape class contains the representation of the cellular landscape model, including parcel definitions. The database stores information relevant to all aspects of the model, including the history of agent interactions and model output.

The following example illustrates only three steps of agent interaction in a land market: (1) put a land parcel on market, (2) get bids, and (3) decide to accept a bid or take the land parcel off the market. In the model, this process would be accomplished as follows (note that <name> indicates a class name):

1. <Agent> puts parcel up for bidding and sends to <Mediator>.
2. <Mediator> sends parcel to <LandMarket>.
3. <LandMarket> requests information on land parcel from <Mediator>.
4. <Mediator> gathers information on parcel from <Landscape> and <Database>.
5. <Mediator> gives information to <LandMarket>.
6. <LandMarket> determines a bid. (This also requires getting input from other agents.)
7. <LandMarket> gives the bid to <Mediator>.
8. <Mediator> gives bid to <Agent>.
9. <Agent> decides to accept or decline the bid using a decision determined with the help of its <AgentDecision> module.

In the example, there are a number of instances of both inheritance and composition. For instance, the Landscape class has instances of the Tree class, as well as the Farmland class. Similarly, the Agent has an AgentDecision class. Subclasses of the AgentDecision class (Reinforcement learning and Bayesian learning) define alternative decision-making mechanisms. Through the use of polymorphism, the correct AgentDecision subclass can be chosen as the program is run. In this case, the mediator only has one agent to which it must give the bid information. However, one could certainly imagine a case where there might be multiple agents who would be interested in such information. In such a case, the mediator, perhaps with the assistance of other classes, would give the appropriate information to the relevant agents.

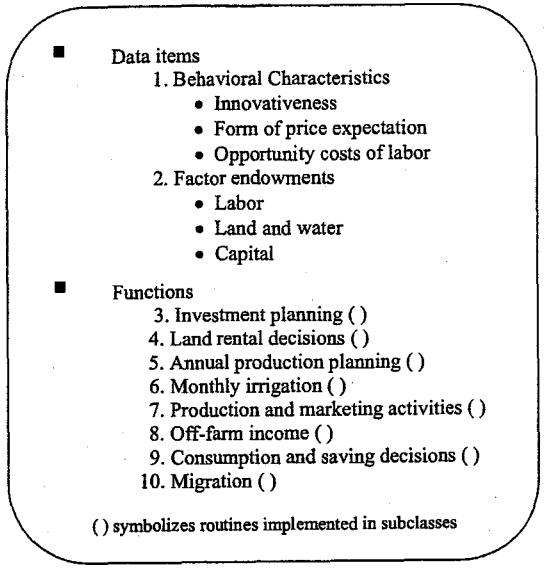

Figure 2. Implementation of a Farm Agent Class

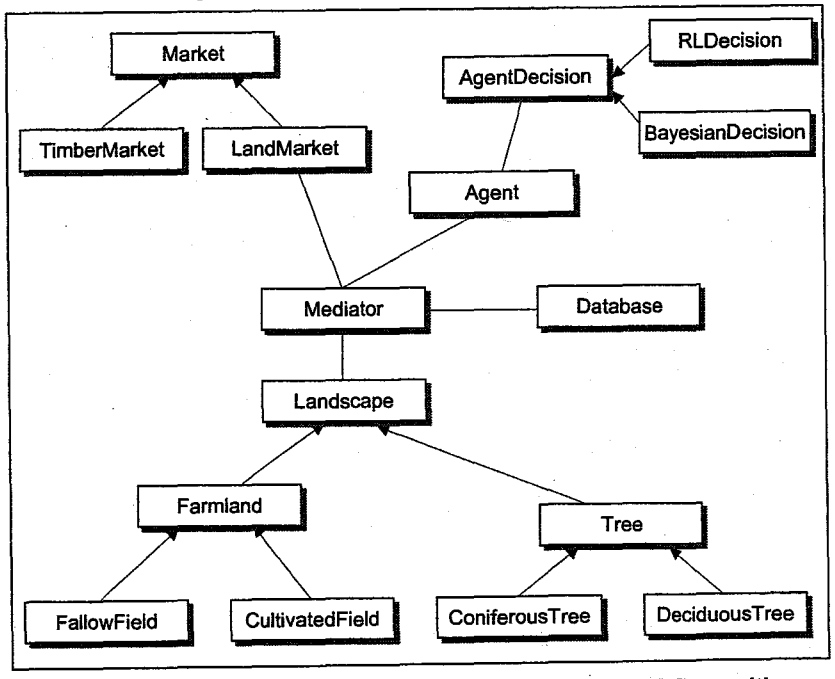

Figure 3. Land Market Example Demonstrating Inheritance and Composition

In OOP, an important consideration is to create classes that are loosely coupled. In loose coupling, information transfer mechanisms between classes are structured independently from the internal functioning of each class. Thus, the information transfer protocol and the internal functioning can operate and be updated independently. The advantage of loose coupling is illustrated in the ability to provide an agent with different decision-making classes without altering any of the code in the agent class or any of the associated classes. The use of the Mediator pattern also contributes to the loosely coupled structure.

Event Sequencing: Synchronous, Asynchronous, and Event-Driven

Workshop participants discussed event sequencing in models. Event sequencing relates to the scheduling of agent decisions and interactions. Two general approaches are possible: predetermined (synchronous or asynchronous) or event-driven. In a synchronous or asynchronous program, agents act in a predetermined manner, even if that might be randomized, as in some asynchronous programs. In an event-driven program, the interaction of agents depends on actions (events) of other agents or of the rest of the computer environment. For example, an agent might react to another agent's decision to sell some timber. Alternatively, an agent might react to an event such as a sudden storm that floods its fields. The main difference between these two approaches is that in event-driven programs, agents' actions follow directly from events in the environment, rather than having each agent action selected in some predetermined manner on each time step.

Combining Pieces

Participants discussed the need for ABM modules to interface with other software tools, GIS in particular. For some simple models, communication with a GIS may not be needed, if the model can be parameterized with an initial landscape and then simply export a final landscape back to the GIS when the model run is complete. However, some cases were identified in which sequential communication between the ABM and GIS might be needed: feedbacks between human actions and the natural environment that relied on GIS modeling (for instance, a vector hydrologic model) and spatial network interactions, especially transportation networks. (For a more detailed discussion of the issue of connecting a GIS and ABM, see Westervelt 2002.) Participants also agreed that integration with a good database program and tools for visual output and display are needed. In principle, a GIS could meet these needs. Finally, in reference to needs for verification and validation, participants stressed the need for either built-in statistical modeling functionality, or seamless linkages with statistical analysis software. In particular, it may be very useful to have standard techniques for validation of landscape models built in to ABM/LUCC tools, as well as a range of non-parametric statistical analysis techniques. There also may be a need for statistical modules that spatially extrapolate aggregate socioeconomic data using standard functions. Access to operations research tools for optimization and analysis of complex systems also may be useful.

Communication and Model Comparisons

Workshop participants noted the need to have mechanisms to communicate the structure, processes, and rules that drive model outcomes. The presentation of a fairly simple model can be quite difficult due to the complex nature of most ABM/LUCC programs (see section 2.3 in Parker *et al.* 2002). There are, however, tools that can help with this difficult task. The Unified Modeling Language (UML) is commonly used to model computer programs. Not only can it show the structure of the code, it also can be used to show interactions between parts of the code, concerns that a particular part of the code might have, the sequencing of events, and more. UML is most commonly used in conjunction with OOP languages, as the two tools were developed concurrently with the express purpose of developing efficient programming practices. A good resource on UML is Fowler and Scott (1999).

Participants identified a set of key model features that should be communicated as part of dissemination of research results, potentially as part of a standardized "model metadata" format. These factors include but are not limited to the number of agents, agent architecture, agent communication, human-biological interactions, the spatial and temporal scales at which the model operates, sequencing and/or event scheduling mechanisms, and frequency and type of updates. The possibility of posting model metadata on an ABM/LUCC research website also was discussed.

Participants discussed the role of comparisons between software models as a part of code verification. One proposed strategy for model comparison would be to have a standardized dataset that a variety of models in different packages are constructed to run against. An alternative strategy would be to have what should be the same process or problem modeled in several languages/platforms. Differences between model outcomes would reveal artifacts.

Object-Based Simulation Platforms

Many specific ABM software platforms were discussed by workshop participants, with an underlying question being whether it would be useful for the community to move to or develop a single, standardized platform that would be widely used for ABM/LUCC. A variety of criteria across which software models could be evaluated were discussed, including:

1. The model's ability to represent space (discrete, continuous, raster, vector) and topological relationships
2. Mechanisms for scheduling and sequencing of events
3. Interoperability with other programs as well as with the Internet
4. Distributed processing capabilities (for speed)
5. Ease of programming and/or using the package
6. Size of the community using that platform
7. Size of programming community familiar with the language in which the package is implemented
8. Ability to represent multiple organizational/hierarchical levels, or scales

One of the major concerns with regard to all types of object-based models is that the results of simulations based on them are difficult to verify. It is difficult to determine if the performance of the computational model is as intended, or if it is due to programming errors or other encoding mistakes. This is discussed in section 2.4. Although it is feasible, the process of error detection is, of course, time consuming and difficult. Thus, to bypass some of these difficulties, and to avoid the need for modelers to keep reinventing the wheel, several groups have developed simulation platforms in which object-based computational models can be implemented. The four described in detail here—SWARM, RePast, Ascape, and CORMAS—are all written in object-oriented programming languages. The review is based on both documentation from the official websites and an informal survey conducted among the developers of the platforms in November 2001.

The SWARM simulation system was originally developed at the Santa Fe Institute and is now maintained by the SWARM Development Group. SWARM is a set of software tools written in Objective-C, an object-oriented language based on C which uses the Smalltalk model of OOP (as opposed to C++, which also is based on C but uses a different model of OOP). Recently it became possible to use Java to call upon the facilities offered by the SWARM libraries. SWARM includes libraries of standard object design and creation routines, analysis tools, and a simulation kernel that supports hierarchical and parallel processing. It is specifically geared toward the simulation of agent-based models composed of large numbers of objects. The sequence in which actions of agents are executed can be sequential, asequential, or event-driven. Event-driven actions indicate that agents react to observed changes rather than just acting at specified times. SWARM is not focused on a particular application, but the large number of users cover many application fields, including geography. SWARM has been a source of inspiration for other ABM platforms like RePast and Ascape, which are more focused on social science applications.

The first version of RePast is mainly based on SWARM but is written entirely in Java. The goal of RePast is to move the representation of agents as discrete, self-contained entities toward a view of social actors as permeable, interleaved, and mutually defining, with cascading and recombinant motives. One of the highlights of RePast is the strong support for network models. As with Swarm, agent actions can be sequential, asequential, or event-driven.

Ascape is being developed at Brookings Institute, the home of the Sugarscape model (Epstein and Axtell 1996), to support the development, visualization, and exploration of agent-based models. It also is written entirely in Java and is designed to be flexible and easy to use. A high-level framework supports complex model designs, while end-user tools make it possible to explore existing models easily. An important difference between SWARM and Ascape is that the latter is simpler to use and has a very complete user interface. Unlike SWARM and RePast, Ascape is not event-driven. In each time step, the agents execute their actions either sequentially or asequentially; Ascape does not allow event-driven scheduling.

CORMAS (Common-Pool Resources Multi-Agent System) is a programming environment dedicated to creating multi-agent systems, with a focus on the domain of natural resources management. CORMAS is being developed at CIRAD in Montpellier, France, and is based on the objective-oriented language Smalltalk. CORMAS provides a framework for developing simulation models of coordination modes between individuals and groups that jointly exploit common-pool resources.

SWARM is the most powerful and comprehensive multi-agent package. A drawback is the steep learning curve since strong programming skills are required. Since SWARM is used in many disciplines, the standard version is not focused on a specific field of application. The other three platforms have shorter histories, and they are more focused on social science applications. Not all tools available in SWARM, such as statistical libraries and GIS connections, are available in RePast, Ascape, and CORMAS. However, these three platforms require fewer programming skills. Each platform has its own focus, which can be characterized as follows: RePast focuses on network dynamics and more comprehensive agents, Ascape concentrates on the ability to create simple models easily, and CORMAS emphasizes the development of applications for common-pool resources together with local stakeholders. A summary of the four platforms is given in Table 1.

2.4. Calibration, Verification, and Validation

Workshop participants noted that as agent-based modeling becomes more common for exploring land-use and land-cover change, they must pay more attention to the challenges of calibration, verification, and validation. Most ABM/LUCC research endeavors have an underlying conceptual model that is implemented as an agent-based model. This model represents a target system that concerns land-use and land-cover change. The model is *calibrated* by fitting it to data before running it. The model is verified by ensuring the proper functioning of its underlying programming. The model is then subject to structural validation (how well the software model represents the conceptual model) and outcome validation (how well model outcomes characterize the target system). Evidence of mounting interest in verification and validation is exemplified by a recent special issue of *Agriculture, Ecosystems and Environment* (Veldkamp and Lambin 2001).

While this section touches on general aspects of calibration, verification, and validation, it is concerned specifically with issues raised by workshop participants about a number of tasks: (1) the potentially problematic relationship between calibration, verification, and validation in terms of data and model fitting; (2) challenges raised by model sensitivity, complexity, and agent interaction; (3) issues surrounding balancing the role of theory and empirical research in structural and outcome validation; (4) problems raised by using spatiotemporal statistics in an agent-based model setting; and (5) the challenges posed by scale, aggregation, and representation.

Data and Model Fitting

A researcher creates a model structure, verifies this structure, calibrates or parameterizes it, and uses it to create outcomes. Data for these tasks are drawn from other models, theories, and observations of the target system provided by surveys, role-playing games, interviews, censuses, and remote sensing. The key challenge with model fitting is that data used for one purpose, such as calibration, should be kept separate from those used for others, such as validation. This need can be at odds with the fact that rarely does data for more than two or three periods exist, given the expense of data acquisition. There is less need to reserve

Table 1. Object-Oriented Packages for Agent-Based Modeling

	SWARM	RePast	Ascape	CORMAS
Developers	Santa Fe Institute/ SWARM Development Group	University of Chicago	Brookings Institute, Washington, D.C.	CIRAD, Montpelier, France
Start development	Early 1990s	Early 1999	1997	1996
Website	http://www.swarm.org	http://repast.sourceforge.net/	http://www.brook.edu/es/dynamics/models/ascape	http://cormas.cirad.fr
Language	Objective C/Java	Java	Java	Smalltalk
Operating System	Unix/Linux, Mac OSX, Windows	Windows, Unix/Linux, Mac OSX	Windows, Unix/Linux, Mac OSX	Windows, Unix/Linux, Mac
Required Experience	Strong skills	Some Java programming	No experience of running existing models, basic skill for changing models, and strong skills to make major extensions	None, if attending the training courses, basic skills in programming otherwise
Event driven?	Yes	Yes	No	No
GIS connection	Kenge/GIS library http://www.gis.usu.edu/swarm/	In development	Beta version	Generic methods to import/export maps from/to MapInfo, both for vector and raster formats. With ArcView, a dynamic link via Access has been successfully tested by using ODBC and DDE
Spreadsheet connection	No	Yes	Yes	Yes
Statistics of runs	The statistical package R, and Splus clone, handles the statistics	User can calculate statistics, the Colt library that comes with RePast provides some statistical functions, and RePast itself can calculate some simple network statistics	Many, like average and variance, Gini coefficient	User can define which data to store
Main focus of applications	Natural and social sciences, military and commercial applications	Social science	Social and economic systems	Economic and ecological simulation, and natural resource management
Available demo-models	On the SWARM website there are only a few demo-models, but there are many journal publications, and a few books with SWARM applications	Six demo-models	About 20–30 demo-models	Numerous models are described on website, with papers and electronic addresses of the authors
Documentation	Yes	Yes	Yes	Yes
Tutorial	Yes	Yes	Rudimentary	Yes
Training	No	There have been courses in the past	No	Various courses are given each year

validation data when the model's structure itself is the "outcome" of interest, as is the case when all available data are used for estimation of regression-based models (e.g., Mertens and Lambin 2000).

Sensitivity, Complexity, and Interaction

Workshop participants noted several means of sensitivity testing and challenges associated with this testing. Model verification lies in forcing the mathematical and computational components of a model to fail by varying model configurations and inputs. Sensitivity testing varies parameters across model runs to ascertain the spatial or temporal limits of a model's applicability as well as finding programming artifacts (Klepper 1997, Stroustrup 1997). Error propagation is estimated from the kinds of operations performed on data (Alonso 1968) or considered in terms of uncertainty within a Monte Carlo framework (Heuvelink and Burrough 1993). These verification methods often rely on assumptions of statistical normality and linearity that can be at odds with models designed to accommodate complex behaviors caused by sensitivity to initial conditions, self-organized criticality, path dependency, or non-linearities (Arthur 1988, Ruxton and Saravia 1998, Manson 2001). There is therefore a need for techniques such as active nonlinear testing, which seeks out sets of strongly interacting parameters in a search for relationships across variables that are not found by traditional verification and validation (Miller 1998).

Validity and Theory

Workshop participants noted that it is unlikely that a completely inductive model will adequately represent causal mechanisms. It is therefore necessary for a model to have close ties to a conceptual framework. A resultant rationale of ABM is that it allows the modeler to express concepts in a number of ways that make it easier to see how the model encodes agent behavior (Kerridge *et al.* 2001). In addition to driving structural validity, theory can be necessary to validate abstract outcomes such as trust or learning. This form of validating is a delicate task; for example, several projects described in this report draw on common-pool resource theory to identify prototypical outcomes that can be compared to model outcomes, but they note that there is growing counter-theoretical experimental evidence. A key reason, therefore, to use agent-based models is the exploration of results at odds with theory, balanced by empirical data.

Statistics, Space, and Time

The complexity of land-use and land-cover change and agent-based modeling demands spatiotemporal tests of outcomes (Turner *et al.* 1989). This is evidenced by applications described in this report that use cross-scale metrics, landscape metrics such as fragmentation; and geostatistical and mathematical measures that identify differences between random "white" noise and others such as "black" and "pink" noise characterized by power density functions. This said, common location-based methods such as error matrix analysis or the kappa statistics have problems that are addressed by newer measures that can differentiate between errors caused by location and by poor quantity estimation (Pontius 2000, Pontius and Schneider 2001). Similarly, while pattern and texture metrics are useful for their ties to

ecological characteristics such as biotic diversity (Giles and Trani 1999), their use is tempered by uncertainty about the linkage between fractal metrics and ecological processes (Li 2000).

Scale and Aggregation

There are scale-related problems held in common by validation, verification, and calibration. A host of statistical issues surrounds the scale at which land-use and land-cover change is observed and modeled (Nelson 2001). Change analysis, for instance, is affected by changing resolution (Lam and Quattrochi 1992) and extent (Saura and Millan 2001) of spatial data. Researchers have known for some time of ecological fallacy (Robinson 1950), modifiable areal unit problem (Openshaw 1977), and aggregation effects (Kimble 1951). These effects can be statistically causal, since variables differentially co-vary as a function of the scale at which they are measured (Bian 1997; Kummer and Sham 1994). Scale issues are also germane to the portrayal of emergent behavior (section 2.2), since emergence can confound a priori aggregation schemes. There are no hard and fast rules for scale-related issues, so the best advice is: "forewarned is forearmed" with continued research (*e.g.*, Chou 1993, Lam and Quattrochi 1992).

Conclusion

Calibration, verification, and validation require use of multiple, complementary methods to identify shortfalls in data, theory, and methodology. These tasks will be aided by better communication of model design through adoption of common languages (section 2.3) and better linkages to other software used in land-use and land-cover change research (*e.g.*, GIS, statistical packages). They should become easier as more attention is paid to theory development and data provision for land-use and land-cover change research.